Architectuur document

Project: PTS2

Project team: Gravity Falls

Teamleden: Asror Wali,

Wout Kamp,

Mehmet Bakirci

Opdrachtgever: Patrick De Beer

Versie: 1.1

Versiedatum: 10 December 2015

Status: Volledig

# Documenthistorie

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Versie** | **Wijzigingen** | **Auteur** | **Datum** | **Verzendlijst** | **Verzoek** |
| 0.1 | Opzet document | Wout Kamp | 30-11-2015 |  |  |
| 0.2 | Toevoeging hoofdstukken 1 t/m 4 | Wout Kamp | 05-12-2015 |  |  |
| 1.0 | Toevoeging hoofdstukken 5 t/m 8 | Wout Kamp | 06-12-2015 |  |  |

Inhoud

Documenthistorie 2

H1 Inleiding 4

H2 Domeinmodel 5

H3 Persistentie 8

H4 Communicatie 11

H5 Realisatie niet-functionele eisen 13

H6 Componenten 14

H7 Deployment 13

H8 Specificatie van interfaces 14

# H1 Inleiding

**Context**

In dit document staan de requirements beschreven van de onze game met portal genaamd ‘Gravity Falls’. Speler kunnen op de portal inloggen met hun eigen account, een lobby in gaan, game room joinen en een spel starten.

**Applicatie**

Betrokken bij dit systeem zijn game servers en cliënten. De actoren in het systeem zijn de klanten en de admins.

**Niet-functionele eisen**

De niet-functionele eisen met betrekking tot betrouwbaarheid, performance, beveiliging en schaalbaarheid.

Betrouwbaarheid: Als iemand disconnect van een game mag de game niet voor andere mensen vastlopen en moet het spel gewoon door blijven gaan.

Performance: De game moet zo min mogelijk lag bevatten en zo snel mogelijk op alles reageren.

Beveiliging: De gebruiker kan na het autoriseren alleen een zijn eigen account inzien.

Schaalbaarheid: Het is mogelijk om 1000 games tegelijk te kunnen spelen.

**Doel van dit document**

In dit document beschrijven we de keuzes die we hebben gemaakt in ons architectuur van systeem en waarom we deze keuzes hebben gemaakt tegen over andere keuzes.

# H2 Domeinmodel

**Klassendiagram**

Hier ga ik mijn klassendiagram toelichten en mijn keuzes verantwoorden. In het Visual Paradigm project vind je het hele klassendiagram terug.

**Game**

In het klassendiagram staat het Cliënt component centraal waar de portalcontroller klassen in is verwerkt. Deze portalcontroller wordt gebruikt om de informatie van de game over te sturen naar alle mensen die op dat moment aanwezig zijn in het spel. Alle spelers krijgen dezelfde informatie overgestuurd van hoeveel spelers er aanwezigt zijn en van die speler wordt dan ook de gebruikernaam weergeven.

Levelstate wordt gebruikt voor het oversturen van alle data van wat er gebeurt in het spel. Alle events worden overgestuurd. Die data van de game bevat bijvoorbeeld: Health, Bullets, Crystal, Movement en Gravity. Deze levelstaat klasse heeft een verbinding met de package ‘socketmsg’. Levelstate gebruikt deze klasse voor het versturen van berichten naar andere cliënten die andere cliënten gebruiken deze informatie om zo hun eigen game te updaten met alle bewegingen van de spelers.

In de server package staat de informatie over de admin van een game. Als een game aangemaakt wordt door een speler wordt hij ook meteen een admin van dat spel. De gegevens van deze admin wordt gebruikt om hiervan een server te starten. Dus elke game heeft zijn eigen server met een ip-adres waar de cliënten mee kunnen verbinden.

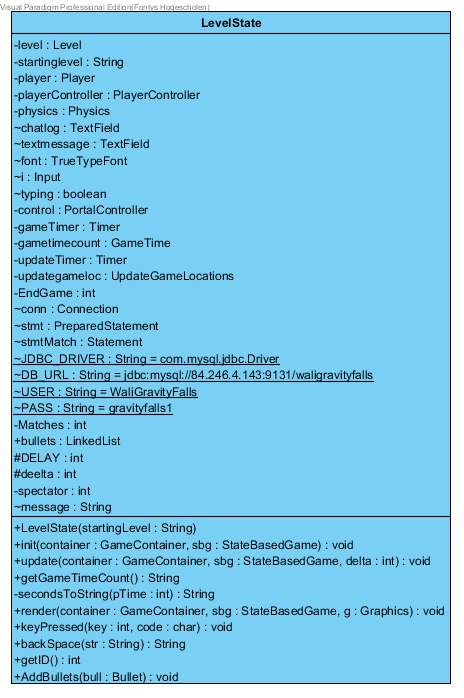
**Portal**

We hebben de portal opgedeelt in twee delen. Het eerste deel is de ‘gravityfallsportal’ en het tweede deel is de ‘gameroom’. In de gravityfallsportal wordt alles geregeld met het aanmaken van nieuwe rooms. Hier kan een speler ook communiceren met alle andere spelers die aanwezig zijn.

Bij de gameroom worden alle spelers ontvangen die mee willen doen aan het spel. In deze room kunnen spelers van de room verwijderd worden en kan het spel gestart worden. Als het spel gestart wordt gaat het door naar de portalcontroller. Bij de gameroom wordt er gebruik gemaakt van het ip-adres van de persoon die de kamer aangemaakt heeft.

# H3 Persistentie

In dit hoofdstuk ga ik de persistentie van objecten beschrijven. Hierin ga ik beschrijven welke eigenschappen van welke objecten moeten worden bewaard en hoe de opslag wordt gerealiseerd. Tot slot schrijf ik hierin ook op wanneer de opslag hiervan plaatsvindt. Dit kan je ook allemaal terugvinden in het Visual Paradigm Project.

****

In levelstate worden een aantal dingen op verschillende manieren opgeslagen.

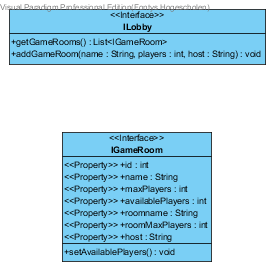
Voor iedere speler wordt op het einde van een game zijn eigen scores opgeslagen.

Deze scores worden opgeslagen in een vooraf aangemaakte database.

Op de portal worden de scores van de speler gebruikt om bij zijn eigen profiel in te laden. Van een speler wordt dan zijn highscore vertoont van al zijn gespeeld games.

Wat er ook op het einde van een game wordt opgeslagen is de game zelf. Deze game wordt ook opgeslagen in een database. Deze game bevat de scores van elke speler en wat standaard waarde. Deze data is voorlopig nergens anders in de applicatie beschikbaar maar alleen maar in de database te zien. De gegevens die van een wedstrijd opgeslagen worden zijn de scores van elke speler en ook de tijdsduur van de gespeelde game.

Lokaal worden ook de kogels opgeslagen die afgevuurd worden door een speler. Deze kogels moet opgeslagen worden voor even om deze op het beeldscherm te laten zien. Deze kogels worden ook overgestuurd naar andere cliënten. Deze kogels worden al na korte tijd weer verwijderd om zo ruimte vrij te kunne maken. Van de kogels worden de positie en de speler die ze afschoot opgeslagen.

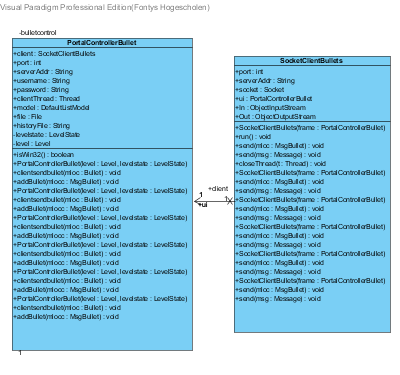


Bij de portal en gamerooms wordt er gebruik gemaakt van twee interfaces. Deze interfaces zorgen voor de lijst met alle gamerooms. Deze gamerooms zijn voor alle spelers te zien. Elke keer als er een speler een gameroom aanmaakt wordt er bij elke andere speler die game room ook toegevoegd.

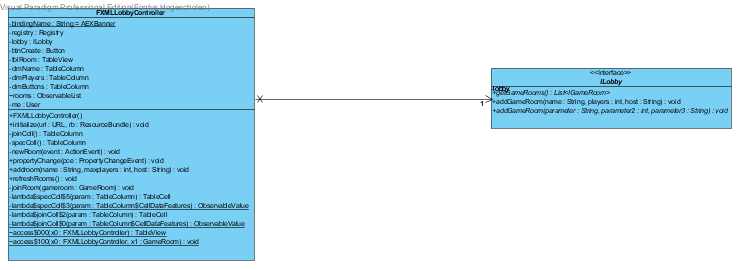
# H4 Communicatie

In dit hoofdstuk ga ik de communicatie van en tussen objecten beschrijven, welke eigenschappen van een objecten er moeten worden gecommuniceerd en hoe dit word gerealiseerd. In de game worden alle data overgestuurd door middel van Sockets en er wordt ook gebruik gemaakt van RMI

**Client component**



Op dit onderdeel maken we gebruik van sockets. Als een speler schiet in een game moet dat voor alle andere spelers te zien zijn. Elke kogel wordt toegevoegd aan een lijst met kogels. In die lijst staat de positie en het id van de speler die schiet toegevoegd. Deze lijst worden dan over en weer gestuurd van speler naar speler.



Hier wordt er gebruik gemaakt van RMI met de connectie tussen de Ilobby en de LobbyController. Dit gebruiken we om alle spelers een update te geven van de aangemaakt gamerooms. De lobby houd bij welke gamerooms er zijn er wordt via een server overgestuurd naar alle cliënts.

# H5 Realisatie niet-functionele eisen

In dit hoofdstuk wordt de realisatie van (overige) niet-functionele eisen besproken. De volgende onderwerpen dienen tenminste te worden besproken: betrouwbaarheid, performance, beveiliging, schaalbaarheid. Eventueel kunnen nog extra niet-functionele eisen worden besproken.

**Performance**

De responsetijd van de applicatie bedraagt maximal 5 seconden. De gebruiker moet zo snel mogelijk zijn taken kunnen afhandelen.

Er moet zo min mogelijk lag zijn in de game.

Ook moeten er tenminste 1000 games gelijktijdig gespeeld kunnen worden.

**Compatibilitieit**

Documentatie van interfaces is beschikbaar om communicatie met andere games mogelijk te maken.

**Bruikbaarheid**

Het systeem kan door 95% van de doelgroep binnen 10 minuten worden gebruikt.

**Betrouwbaarheid**

Wat voor de performance erg belangrijk moet zijn is dat het systeem niet meer dan 1 keer mag uitvallen voor 1 uur binnen 24 uur.

Dit zou een belangrijk punt voor de performance zijn.

Als je wil dat mensen jou applicatie blijven gebruiken moet je zorgen dat het ook blijft werken voor 99% van de tijd.

**Beveiligbaarheid**

De gegevens van een gebruiker moeten via een beveiligde verbinding verstuurd worden.

Wachtwoorden van gebruikers worden versleuteld opgeslagen, zodat er niet zomaar aan deze wachtwoorden gekomen kan worden.

De gebruiker moet zich vetrouwd voelen bij het gebruiken van de applicatie.

**Onderhoudbaarheid**

Het systeem is modulair opgebouwd.

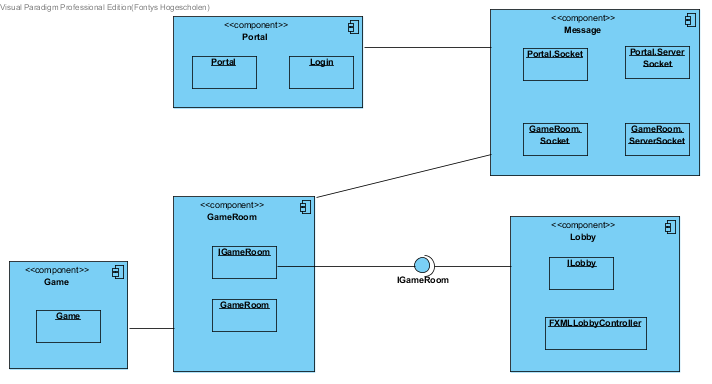
De source code is geschreven volgens een erkende coderingsstandaard, zodat deze code hergebruikt kan worden.

Documentatie van de source code is beschikbaar dit ondersteund en verhelderd de code op.

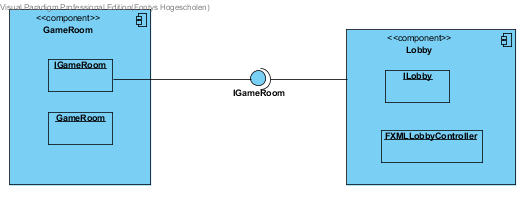
# H6 Componenten

In dit hoofdstuk wordt de opdeling in software componenten besproken aan de hand van een Componentendiagram met toelichting. Daarnaast wordt koppeling en synchronisatie tussen de componenten besproken en de opdeling in software packages.

**Componentendiagram**

****

De Portal component en de GameRoom component gebruiken allebei sockets. Deze twee componenten maken dan ook gebruik van het component message. De gameroom maakt gebruik van de gameroom sockets en de portal sockets maakt gebruik van de portal sockets.



Voor de Lobby gebruiken en de gameroom gebruiken wij RMI. Deze RMI is voor het oversturen van de rooms die wij kunnen aanmaken. De Ilobby krijgt een interface van Igameroom waar de info staat over de gameroom zelf. De Lobby en de Lobbycontroller gebruiken de RMI om lobbycontroller aan te passen als er bij de Ilobby een gameroom wordt toegevoegd.

Elke keer als er een gameroom aangemaakt door een speler moet er voor de andere spelers aangegeven worden of het aantal rooms veranderd is. Als de rooms inderdaa veranderd zijn wordt dit bij de spelers aangepast.

**Koppeling tussen componenten**

De koppeling die zal plaats vinden twee aantal componten doormiddel van RMI.

De interface Ilobby maakt gebruik van RMI om zo rooms te kunnen weergeven op de lobbycontroller.

**Remote objecten**

De remote objecten die er worden gebruike is de interface van de Ilobby.

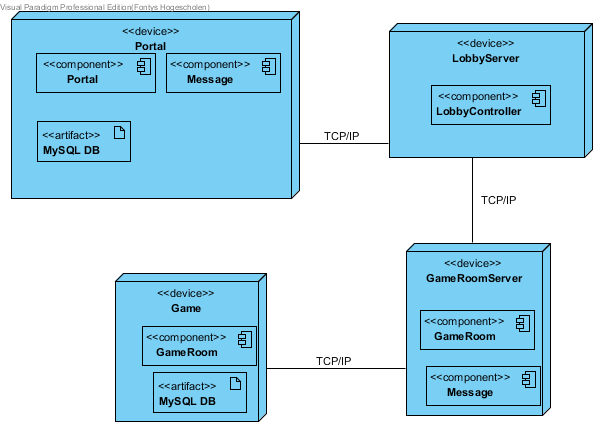
Al deze interfaces worden op afstand aagesproken door andere componenten.

Deze Ilobby hoeven we maar voor een ding te gebruiken en dat is voor het oversturen van de lobby’s

# H7 Deployment

In dit hoofdstuk wordt de toekenning van software componenten aan hardware besproken aan de hand van een Deploymentdiagram.

**Deploymentdiagram**



We heb ervoor gekozen om alle componenten apart op een server te draaien en de database op dezelfde server te laten draaien. Verder zijn alle componenten via TCP / IP verbonden.

# H8 Specificatie van interfaces

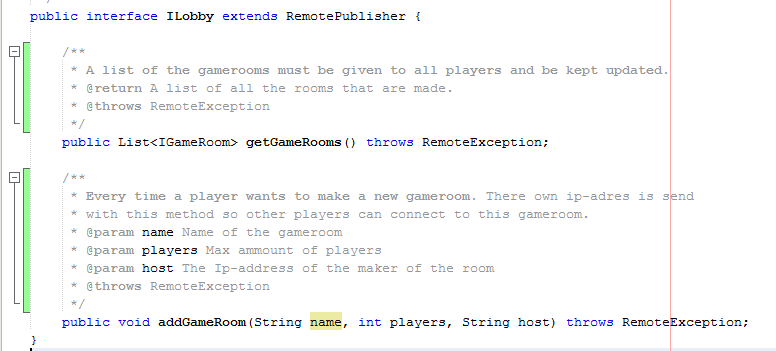
In dit hoofdstuk wordt de specificatie van interfaces besproken. Voor ieder interface wordt per methode gedefinieerd:

* Naam methode
* Naam en type argumenten
* Precondities
* Type returnwaarde
* Beschrijving

De volgende interfaces worden besproken:

* ILobby
* IGameRoom

**ILobby**



**IGameroom**

